VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Simulační prostředí OMNeT++

Simulation environment OMNeT++

2009 Michal Zdařil

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2009 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nikdy nevznikla. Především bych chtěl poděkovat mojí rodině, za její neustálou podporu a víru v mé schopnosti. Dále pak vedoucímu mé práce panu Ing. Janu Vavříčkovi, bez jeho trpělivosti a pomoci by to také nešlo. Nakonec všem přátelům a známým, kteří podali pomocnou ruku ať už jen formou povzbudivého slova, či formou jakékoliv jiné pomoci.

Abstrakt

Cílem práce je poskytnout zájemci o simulační systém OMNeT++ stručný pohled do této problematiky. V první kapitole je čtenář seznámen se základními pojmy diskrétní událostní simulace a hlavními pojmy z oblasti simulačního prostředí. Následuje popis instalace krok za krokem ať už OMNeTu++ jako takového, či začlenění INETMANET frameworku do vývojového prostředí Eclipse. Další část textu poskytuje podrobný pohled na hlavní rozšíření OMNeT++ a to INETMANET framework, který obsahuje moduly určené pro simulaci počítačových sítí. Závěrečná kapitola si neklade za cíl popsat veškeré možnosti, které nabízí INETMANET, ale věnuje se tvorbě testovací topologie, jež nám ukáže základní schopnosti systému a způsob, jak efektivně budovat své vlastní topologie.

Klíčová slova: OMNeT++, INETMANET, simulační prostředí, modul, Eclipse, .NED, .INI

Abstract

The purpose of this thesis is to provide short view to OMNeT++ discrete event simulation system. In the first chapter the discrete simulation system and main concepts of OMNeT++ are introduced to a reader. Next chapter describes installation step-by-step of the OMENeT++ and explains how to get the combination OMNeT++ version 4 and INETMANET framework working into integrated development environment Eclipse. Next part of text is focused in detail on INETMANET framework, which extends the main system with extra modules designated for simulate computer networks. Final chapter doesn't try to describe all functions of the INETMANET framework, but explain how to create test topology, which shows us the usage of system and technique of creating own topologies.

Keywords: OMNeT++, INETMANET, simulation environment, module, Eclipse, .NED, .INI

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

S rostoucí složitostí počítačových sítí a množstvím různorodých síťových zařízení, jejichž cena není ani v dnešní době zrovna malá, se jeví jako rozumné otestovat počítačovou síť na některém z dostupných simulačních prostředí. Tyto simulátory umožňují testovat různé technologie a děje, které by na skutečných zařízeních nebylo možné otestovat, nebo by to bylo příliš nákladné. Jako příklad můžeme uvést testování nových síťových protokolů, popřípadě simulaci různých DoS útoků, apod.

Stejně tak v oblasti výuky počítačových sítí, která se dnes stále rozšiřuje na více škol. Ne však všechny školy si mohou dovolit investovat nemalé finanční prostředky do výstavby moderní síťové učebny. Proto jistě uvítají možnost simulace WAN a LAN sítí s technologiemi jako ATM, TCP, IP, Ethernet, Token ring apod. K dispozici je široká paleta simulačních prostředí od těch jednodušších, které umožňují vytvářet síťovou topologii, specifikující síťové uzly, spojení mezi uzly a také provoz mezi těmito uzly, až po ty složitější, které nabízí uživateli možnost specifikovat téměř vše, co se týká protokolů, jenž řídí síťový provoz.

Tato práce se bude dále zabývat diskrétním simulačním prostředím OMNeT++. Je kompletně napsaný v C++ a jeho využití je především v simulaci počítačových sítí. Díky jeho flexibilní architektuře se však dá použít i k simulaci IT systémů, hardwarové architektury a distribuovaných systémů. Simulační modely se skládají z modulů, které jsou hierarchicky uspořádané. Komunikace mezi moduly je založena na bázi předávání zpráv. Popis topologie je vytvářen za pomoci jazyka NED (Network Description). OMNeT++ disponuje silným grafickým rozhraním, které je odvozeno od Eclipse. Ten mimo jiné umožňuje graficky upravovat topologii sítě, krokovat, spouštět animovanou simulaci, debugovat, apod.

Je šířen pod licencí GNU GPL pro akademické a nekomerční použití. Pro komerční využití je k dispozici placená verze OMNEST. Právě proto se těší velké oblibě rozsáhlé komunity ať již koncových uživatelů, nebo těch, kteří přispívají svými nápady při tvorbě nových modelů, či udržování stávajících. Na oficiálních stránkách projektu [3] je také možno najít podrobný manuál [1], uživatelskou příručku [2] a několik výukových videí a tutoriálů. Funguje také komunitní Wiki [6] a fórum.

OMNeT++ byl vyvinutý Andreasem Vargou na půdě technické univerzity v Budapešti. Současná verze OMNeT++ 4.0 je k dispozici pro platformy Linux a Windows (XP a Win2K).

K základnímu OMNeT++ jako takovému jsou k dispozici ke stažení přídavné frameworky, které obsahují kolekci modulů určenou pro specifickou simulaci. Např.: INETMANET – zabývající se simulací počítačových sítí, ReaSE – rozšíření INET frameworku o možnost simulovat reálné síťové útoky, IPv6Suite – rozšíření INETu pro podporu IPv6 protokolu a další.

My se zde budeme zabývat především INETMANET frameworkem. Jde o kolekci modulů speciálně určených pro simulaci počítačových sítí. Obsahuje modely pro TCP, IP, UDP, PPP, Ethernet, MPLS a další protokoly. Navazuje na starší verzi INET framework, který byl rozšířen o podporu mobilních a bezdrátových simulací.

V závěrečné kapitole se podíváme na ukázkový příklad zde popisovaného frameworku. Ukážeme si tvorbu testovací topologie tak, abychom co nejefektivněji využili možnosti poskytované robustním prostředím.

Tento text se snaží podat stručně a srozumitelně všechny potřebné informace pro ty, kdo by chtěli s tímto systémem a příslušnými frameworky pracovat, aniž by museli zkoumat několika set stránkový manuál.

2 Základní vlastnosti a charakteristiky

V této kapitole si popíšeme simulační systém OMNeT++. Vysvětlíme si základní principy a struktury a podíváme se podrobněji na jazyk NED.

2.1 Úvod

OMNeT++ je diskrétní událostní simulační prostředí primárně určené pro simulaci počítačových sítí, ale vzhledem k jeho flexibilní architektuře se využívá v oblasti různých IT systémů a obchodních procesů. Jeho autorem je Andreás Varga, který jej vytvořil na technické univerzitě v Budapešti. Je psán v jazyce C++, šířen pod licencí GNU GPL (General Public Licence) a tudíž je pro nekomerční využití k dispozici zcela zdarma.

OMNeT++ je založen na modulární architektuře, díky níž dovoluje pomocí hierarchicky vložených základních modulů vystavět téměř libovolné simulační modely. Silné grafické rozhraní, kvalitní online dokumentace a především dobrá škálovatelnost se staly hlavními důvody pro rozšíření zejména v univerzitním a výzkumném prostředí.

2.2 Diskrétní simulace

Simulační systémy mají za úkol reprezentovat skutečné systémy pomocí vhodného zjednodušení. Zjednodušení reálného systému se nazývá model, který se skládá z menších prvků. Tyto prvky mají své vlastnosti, jejichž změnou dojde k vytvoření události. Diskrétní simulace je reprezentována jako chronologická sekvence událostí, na jejichž vznik daný systém vhodně reaguje.

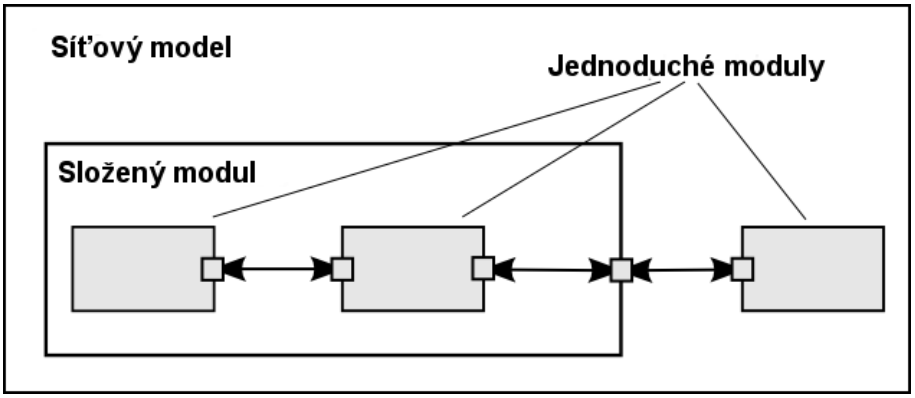
V simulačních systémech rozeznáváme 2 druhy času výskytu události:

- Simulační (simulation time) – čas v modelu
- Reálný – čas CPU – udává, kolik strojového času spotřeboval

U diskrétních simulačních systémů nehraje čas mezi jednotlivými událostmi žádnou roli a je tedy možné ho přeskočit. Simulační systém uchovává všechny události v datové struktuře, jenž se nazývá FES (Future Event Set). Takový systém se zpravidla potom chová, jak naznačuje následující pseudokód:

```
while ( není konec simulace )
{
   vyjmi 1 událost z FES;
   nastav simtime_t  t = hodnota času právě vyjmuté události;
   proveď danou událost ( handle Message příslušného modulu);
   důsledek: modifikace modelu + případné naplánování dalších událostí;
}
```

Výpis 1: Zpracování událostí



Obrázek 1: Princip zanořování modulů

2.3 Moduly

Modely jsou složeny z hierarchicky vkládaných modulů, jejichž hloubka zanoření není nikterak omezena. To dává k dispozici mocný nástroj pro tvorbu libovolně komplikované systémové struktury. Nejvýše postavený modul je tzv. systémový. Ten se může skládat ze složených (compound) a jednoduchých (simple) modulů, jak je vidět na obrázku 1. Složené moduly se mohou dále dělit, až na konci narazíme na jednoduchý modul, který je nedělitelný a vykonává určitou funkci.

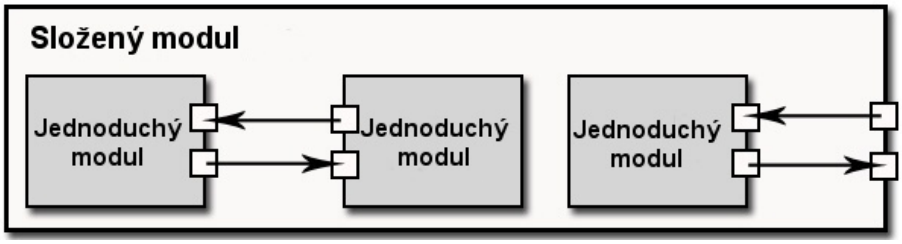
Oba typy modulů dědí ze třídy cModule. Při použití jako stavební blok konkrétního modelu není třeba rozlišovat, zda je to jednoduchý či složený typ modulu. To umožňuje uživateli pracovat s již hotovými moduly transparentně; rozdělit jednoduchý modul do několika zapouzdřených v složeném modulu, či naopak, přeimplementovat funkci složeného modulu do jednoho jednoduchého modulu bez ovlivnění funkčnosti použitých modulů pro jiné uživatele.

2.3.1 Jednoduchý modul

Jednoduché moduly jsou základním stavebním kamenem a jsou atomickými (dále nedělitelnými) prvky, které jsou naprogramovány v jazyku C++ za pomoci simulační knihovny OMNeT++. Tyto prvky generují a reagují na události, čímž dávají těmto modulům funkčnost.

2.3.2 Složený modul

Složené moduly slouží pouze k vhodnému logickému uspořádání systému. Tyto moduly mohou mít parametry a brány stejně jako jednoduché moduly, ale nechovají se aktivně – negenerují žádné události, apod.



Obrázek 2: Princip spojování modulů

2.3.3 Zprávy a spojování modulů

Jednotlivé moduly spolu komunikují pomocí zasílání zpráv. Zpráva může reprezentovat například rámec nebo paket, pokud se jedná o simulaci počítačové sítě, nebo se může jednat o nějakou vedlejší informaci. Jednotlivé moduly mohou zasílat zprávu přímo cílovému modulu nebo mohou definovat cestu pomocí bran. Brány slouží jako vstupní a výstupní rozhraní modulů. Spojením dvou bran vznikne cesta mezi jednotlivými moduly. Spojovat bránami lze buď dva sousední moduly, nebo synovský s rodičovským, jak naznačuje obrázek 2.

Z hierarchické struktury vyplývá, že cesta zprávy je daná posloupností spojů a pomocí bran se dostane z jednoho jednoduchého modulu do druhého. Jednotlivým spojením se dají přiřadit atributy jako je chybovost, zpoždění a rychlost přenosu dat.

2.3.4 Jazyk NED

Jazyk NED (Network Description) slouží k popisu topologie jednotlivých modulů a k podrobné specifikaci struktury celé sítě. Typickými součástmi jsou prostředky pro deklaraci jednoduchých modulů, definici složených modulů a sítí. Tyto soubory mají většinou koncovku `.ned`. Definované moduly lze znovu použít za pomocí klíčového slova `import`.

***import** „název modulu“;*

Definice komunikačních kanálů umožňuje jednotlivým spojením nastavit určité parametry. Každému spojení lze nastavit `delay` (zpoždění), `data rate` (rychlost) a `error` (chybovost).

```
channel <jméno> extends ned.DatarateChannel
{
    datarate = 100Mbps;
    delay = 100us;
    ber = 1e−10;
}
```

Výpis 2: Definice komunikačních kanálů

Jednoduché moduly jsou základními prvky modelu. Definice je uvozena klíčovým slovem `simple` a tělo je uvedeno ve složených závorkách. Uvnitř definice se nachází

klíčové slovo `parameters`, za kterým následuje výčet parametrů spolu s jejich hodnotami a klíčové slovo `gates`, které specifikuje popis bran a jejich spojení.

```
simple <jméno modulu>
{
    parameters:
        // seznam parametrů
    gates:
        // výčet bran
}
```

Výpis 3: Definice jednoduchého modulu

Složené moduly jsou uvozeny klíčovým slovem `module`. Stejně jako jednoduché moduly mají v těle seznam parametrů a výčet bran. Oproti jednoduchým modulům zde můžeme ještě najít specifikaci podmodulů, ze kterých se daný modul skládá a seznam spojení mezi nimi. Parametry složených modulů se většinou předávají do jednoduchých modulů, nebo slouží k dynamickému vytváření struktury. Například aktivní síťový prvek, jehož parametr může být počet portů a podle tohoto parametru se dynamicky vytvoří vstupní a výstupní brány.

```
module <jméno modulu>
{
    parameters:
        //...;
    gates:
        //...;
    submodules:
        // vnořené moduly
        <jméno instance>: <typ modulu> {
            parameters:
                //...;
            gates:
                //...;
    connections:
        // spojení v rámci modulu
        <jméno instance>.<jméno brány> --> <jméno instance>.<jméno brány>;
}
```

Výpis 4: Definice složeného modulu

Na vrcholu celé hierarchie stojí síťový modul, který slouží k sestavení sítě z jednotlivých modulů. Struktura je podobná jako u složeného modulu. Je třeba specifikovat název sítě, popřípadě lze přiřadit parametry.

```
network <jméno modulu>
{
    parameters:
        numUsers=10;
        httpTraffic =true;
    types:
        //...;
    submodules:
        <jméno instance>: <typ modulu> {
```

```
        parameters:
            //...;
        gates:
            //...;
    }
    modul2: Module2 {
        //....;
    }
connections:
    //...;
}
```

Výpis 5: Definice systémového modulu

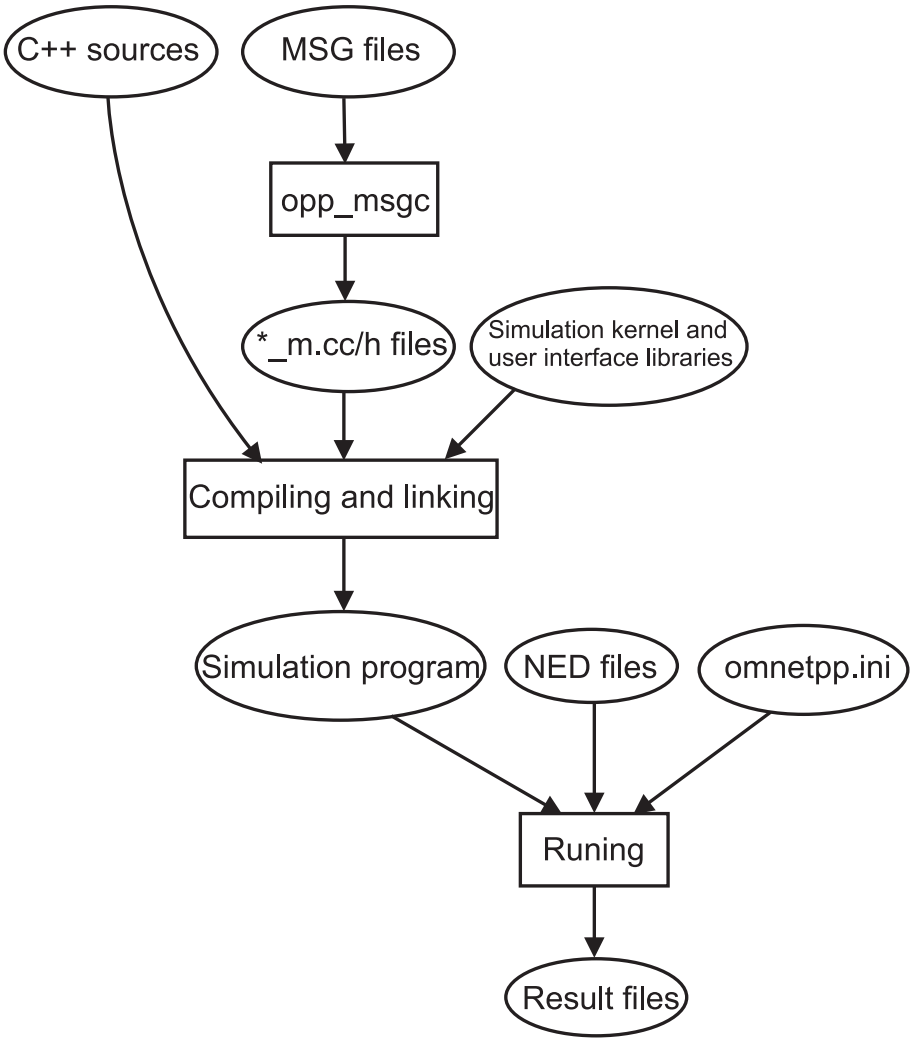
2.4 Simulace

Simulační model se skládá z těchto částí:

- NED soubor – popis struktury topologie
- Definice zpráv – koncovka .msg
- Zdrojové kódy jednoduchých modulů v C++

Konfigurace simulace se nachází v souboru `omnetpp.ini`. Simulaci lze pouštět pomocí příkazové řádky, nebo přes grafické uživatelské rozhraní, které poskytuje názornější a komfortnější práci.

Na obrázku 3 je názorně vidět fáze sestavení a spuštění simulace.



Obrázek 3: Sestavení a spuštění simulace

3 Instalace simulátoru a frameworků

V této kapitole si popíšeme samotnou instalaci OMNeT++ jak pro Windows, tak pro Linux (a kompatibilních systémech). Dále si ukážeme, jak instalovat další frameworky.

3.1 Úvod

OMNeT++ je k dispozici pro Windows, Linux a pro systémy založené na Unixu jako např.: FreeBSD, Mac OS X, aj. V současnosti se vyskytuje ve verzi 4.0 a je možno jej stáhnout z oficiální stránky projektu `http://www.omnetpp.org`. V případě komplikací vzniklých při instalaci je zde možné také najít pomoc přímo na hlavních stránkách [3], na fóru, nebo na komunitní Wiki [6].

3.2 Linux

3.2.1 Instalace

Stáhneme si balíček `omnetpp-x.x-src.tgz` (x.x označuje číslo verze) z oficiální stránky projektu.

Vytvoříme si kdekoliv adresář (v našem příkladě `omnet`) a rozbalíme.

```
~/> mkdir omnet
~/> cd omnet
~/omnet /> tar zxvf omnetpp-<verze>-src.tgz
```

Tímto se nám vytvoří podadresář `omnetpp-<verze>` a v něm rozbalené soubory. Nyní je ještě potřeba příkazem `export` přidat cestu do podadresáře `bin` do systemové proměnné `PATH`, která obsahuje prohledávané adresáře (pro spouštění programu bez uvedeni cesty).

```
export PATH=$PATH:~/omnet/omnetpp-<verze>/bin
```

Po tomto příkazu bychom měli restartovat příkazovou řádku. Učiníme tak odhlášením a opětovným přihlášením.

K překladu a běhu OMNeT++ jsou zapotřebí následující balíčky:

`build-essential, gcc g++, bison, flex, perl, tcl8.4, tcl8.4-dev, tk8.4, tk8.4-dev, blt, blt-dev, libxml2, libxml2-dev, zlib1g, zlib1g-dev, libx11-dev`

Jejich instalaci provedeme příkazem:

```
sudo apt-get install <název balíčku>[1]
```

Nyní se už zbývá přesunout do rozbaleného adresáře a provést překlad souborů.

[1] Týká se jen distribucí používající DEB balíčky. V jiných distribucích je potřeba použít odpovídající nástroj.

```
~/omnet /> cd omnetpp-<verze>
./configure
make
```

Pro otestování úspěšnosti instalace spustíme jeden z ukázkových příkladů.

```
cd samples/dyna
./dyna
```

3.2.2 Spuštění IDE

Grafické rozhraní je od verze 4.0 založeno na vývojovém prostředí Eclipse. Pro jeho bezproblémový chod je zapotřebí `sun-java5-jre`, nebo novější. Spustit lze pomocí příkazu `omnetpp`.

IDE podporuje pouze následující platformy:

- Windows 32bit
- Linux 32/64bit (i386)
- Mac OS X 10.4/5 (i386/ppc)

3.3 Windows

3.3.1 Instalace

Stáhneme si OMNeT++ určený pro Windows z oficiální stránky projektu. Zip archiv obsahuje `MSYS`, `MINGW` a další potřebné součásti jako např.: `perl`, `tcl/tk`, `libxml2`, atd., které slouží pro překlad simulačního systému v prostředí Windows. Jediné, co bychom měli instalovat zvlášť, je `Java Runtime Environment v 1.5` a vyšší.

Archiv přesuneme na místo, kde budeme chtít instalovat a rozbalíme.

```
unzip omnetpp-<verze>-src-windows.zip
```

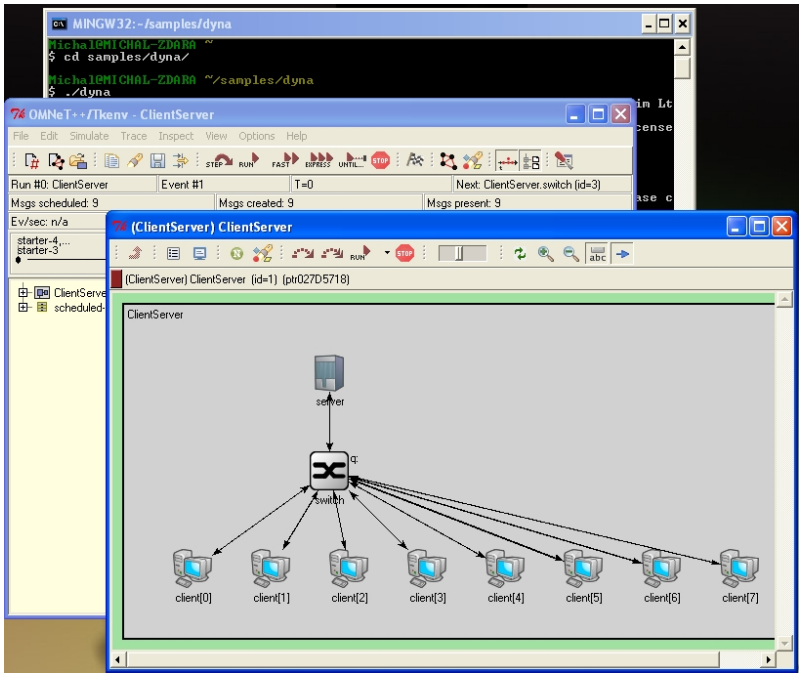
Tím se nám ve zvoleném adresáři vytvoří podadresář `omnetpp-<verze>` a v něm všechny potřebné soubory pro překlad. Přesunem se tedy do tohoto adresáře a zadáme příkaz `mingwenv.cmd` a tím se nám spustí `mingw` prostředí.

Nyní je již situace stejná jako v Linuxu. Zkontrolujeme, zda máme vše potřebné pomocí příkazu `./configure` a následně provedeme překlad obligátním `make`.

```
./configure
make
```

Nakonec provedeme otestování tím, že spustíme některý z příkladů.

```
cd samples/dyna
```



Obrázek 4: Ukázka grafického uživatelského rozhraní

```
./dyna
```

Po spuštění by se mělo objevit grafické uživatelské rozhraní, které je vidět na obrázku 4. Zde je také ukázáno prostředí `mingw`.

3.3.2 Spuštění IDE

Pro běh je zapotřebí `Java Runtime Environment 5.0` nebo vyšší. Spouští se příkazem `omnetpp`.

3.4 Frameworky

Frameworky rozšiřují základní OMNeT++ o další moduly. Každý framework má určité zaměření, ať už se jedná o oblast počítačových sítí v případě `INETMANET`, či třeba `File System Simulation` zabývající se souborovými systémy. Více o frameworcích pojednává další kapitola.

3.4.1 Instalace

Samotná instalace se skládá ze stažení zdrojových souborů, jejich rozbalení a následným importem těchto souborů do IDE.

3.4.2 Importování INETMANET frameworku do OMNeT++ IDE

Na začátku je potřeba mít funkční OMNeT++ 4.0. Rozbalíme archiv se soubory.

Otevřeme si IDE a naimportujeme INET projekt:

```
File --> Import --> General --> Existing Projects into Work-
space --> Select <INETMANET> --> Finish
```

Celý balík přeložíme pomocí `ctrl+b`.

Nyní už jen stačí vytvořit nový projekt a ve vlastnostech na záložce `project refe-rences` zaškrtnout INET. Nyní by nám měli být k dispozici všechny prvky INETMANET frameworku.

4 Základní vlastnosti simulačního modelu INET framework

V této kapitole si probereme podrobněji INETMANET framework a zaměříme se i na další frameworky určené nejen k simulaci počítačových sítí.

4.1 INETMANET framework

INETMANET framework je open-source balík obsahující kolekci modulů určených pro simulaci počítačových sítí. Jsou zde k dispozici moduly implementující základní protokoly TCP, UDP, IPv4, IPv6 a další. Dále zahrnuje MPLS, PPP a Ethernet. Od léta roku 2005 přebral INET framework mobilní a bezdrátové moduly z Mobility frameworku a dostal jméno INETMANET. Další vývoj Mobility frameworku byl ukončen a všechen kód byl integrován do zcela nového projektu MiXiM, jehož vývoj je teprve v počátcích.

Síťové protokoly jsou modelovány za pomocí jednoduchých modulů, které jsou dále kombinovány pro vytvoření modelů směrovačů, přepínačů, či pracovních stanic. Síťová rozhraní jsou složené moduly skládající se z fronty požadavků a jednoduchého modulu daného protokolu. Dále jsou zde ostatní moduly zastávající jiné funkce jako např.: směrovací tabulka, tabulka rozhraní, nebo konfigurátor sítě.

4.1.1 Struktura INET a jeho moduly

Popíšeme si adresářovou strukturu balíku, která přibližně odpovídá jednotlivým vrstvám ISO/OSI modelu.

`examples/` Ukázky příkladů využívající možnosti balíku.

`nodes/` Modely zařízení založených na IP, IPv6 a MPLS.

`applications/` Obsahuje modely aplikační vrstvy. Jako například utilitu ping, generátor rámců pro ethernet nebo IP datagramů.

`transport/` Popis transportních protokolů TCP a UDP.

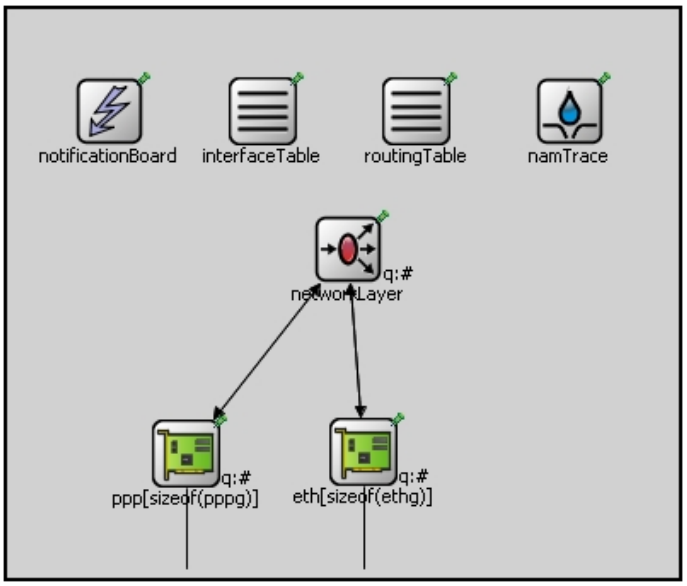
`network/` Obsahuje sadu protokolů síťové vrstvy – IPv4, IPv6, MPLS, LDP, ARP. Dále pak modely pro autokonfiguraci IP a routování.

`networkinterfaces/` Obsahuje modely linkové vrstvy – Ethernet(MAC, LLC, Encap), základní PPP, 802.11b ad-hoc model.

4.1.2 Ukázka modelu směrovače

Pro pochopení základních principů si nyní rozebereme model směrovače. Podíváme se v rámci zjednodušení na jeho reprezentaci v GUI. Model reprezentující směrovač je vidět na obrázku 5.

Model se skládá ze složených modulů síťové vrstvy (Network Layer), vstupně-výstupních rozhraní pro příjem a odesílání zpráv PPP a Ethernet.



Obrázek 5: Model směrovače

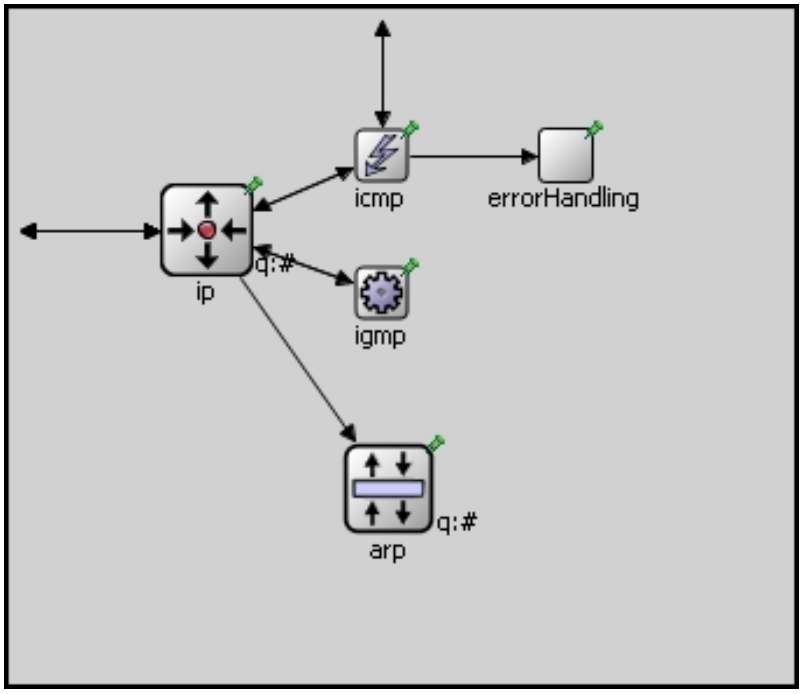
networkLayer – na obrázku 6 můžeme vidět složení tohoto modulu.
Obsahuje tyto jednoduché moduly:

- **IP** – slouží jako rozhraní mezi 3 a 4 vrstvou ISO/OSI
- **ICMP** – implementuje ICMP protokol a stará se o vytváření a zpracovávání zpráv jako např.: echo, reply, time exceeded, destination unreachable, aj.
- **ARP** – implementuje ARP protokol
- **IGMP** – implementuje igmp protokol

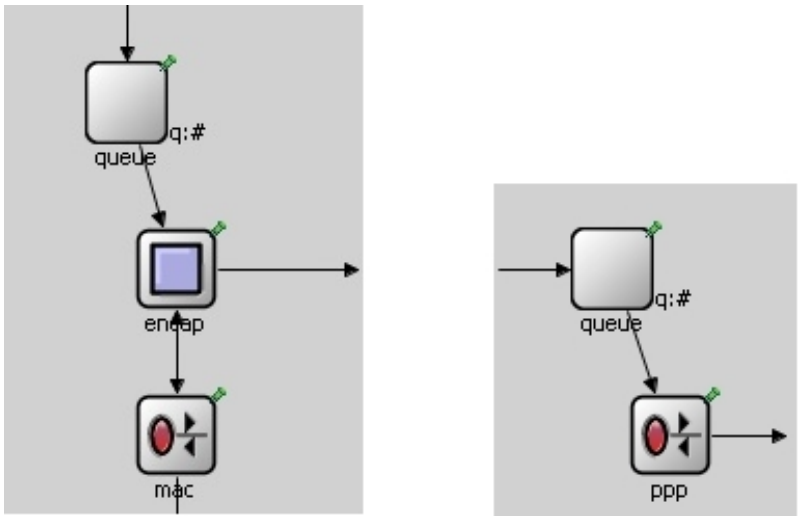
Moduly PPP a ethernet ilustruje obrázek 7. Jak je z tohoto obrázku patrné, moduly se skládají z fronty a příslušného jednoduchého modulu protokolu (PPP nebo EtherMAC a EtherEncap).

Model směrovače dále obsahuje jednoduché moduly z obrázku 5:

- **notificationBoard** – jedná se o modul, který informuje ostatní moduly např.: o změnách směrovací tabulky, stavu vstupně výstupních rozhraní (up/down), změny v konfiguraci rozharní, aj. Tento modul funguje jako prostředník mezi modulem, kde změna vznikla a moduly, které tato změna zajímá.
- **namTrace** – nejméně důležitý modul, zapisuje pouze informace, které obdržel od NotificationBoard
- **interfaceTable** – uchovává seznam rozhraní směrovače



Obrázek 6: Model síťové vrstvy



Obrázek 7: Moduly Ethernet a PPP rozhraní

- **routingTable** – směrovací tabulka pro uložení směrovacích informací. Tyto informace ovlivňují nastavení vstupně výstupních rozhraní. Směrovací tabulka je načtena ze souboru, jehož cesta je definována v NED souboru:

```
string rFile = default(""); //jméno souboru směrovací tabulky
```

Soubor se směrovacími informacemi:
Tento soubor má koncovku .irt nebo .mrt a může obsahovat konfiguraci síťových rozhraní a statické cesty.

```
ifconfig :

# ethernet card 0 to router
name: eth0  inet_addr: 172.0.0.3   MTU: 1500 Metric: 1  BROADCAST MULTICAST
Groups: 225.0.0.1:225.0.1.2:225.0.2.1

# Point to Point link 1 to Host 1
name: ppp0  inet_addr: 172.0.0.4   MTU: 576 Metric: 1

ifconfigend.

route:
172.0.0.2   *           255.255.255.255 H 0   ppp0
172.0.0.4   *           255.255.255.255 H 0   ppp0
default:    10.0.0.13   0.0.0.0         G 0   eth0

225.0.0.1   *           255.255.255.255 H 0   ppp0
225.0.1.2   *           255.255.255.255 H 0   ppp0
225.0.2.1   *           255.255.255.255 H 0   ppp0

225.0.0.0   10.0.0.13   255.0.0.0       G 0   eth0

routeend.
```

Výpis 6: Příklad výpisu souboru obsahující směrovací informace

`ifconfig...ifconfigend.` – část pro konfiguraci síťových rozhraní
`route..routeend.` – část pro konfiguraci statických cest

Parametry rozhraní:

- `name:` – libovolný název rozhraní (např.: eth0, ppp0)
- `inet_addr:` – IP adresa
- `Mask:` – maska
- `Groups:` Multicastové skupiny.
- `MTU:` – např.: Ethernet: 1500
- `Metric:` – metrika

- `flags`: BROADCAST, MULTICAST, POINTTOPOINT

Parametry statických cest:

`Destination, Gateway, Netmask, Flags, Metric, Interface`

`Destination`, `Gateway` a `Netmask` mají běžný význam. `Destination` může být IP adresa nebo „default:“ (defaultní cesta). `Gateway` – * má stejný význam jako 0.0.0.0.

Flag označuje typ záznamu:

- `H "host"`: přímá cesta (přímo připojená ke směrovači)
- `G "gateway"`: vzdálená cesta (dosažitelné skrze některý další směrovač)

`Interface` – jméno rozhraní např.: `eth0`.

4.2 Další framework

Přestože jsme si nejen v úvodu říkali o možnostech OMNeT++ simulovat i jiné oblasti nežli jenom počítačové sítě, je v dnešní době podpora takových projektů na nízké úrovni a mnohdy se jedná o projekty staré i více než 5 let a již neudržované. Proto si zde krátce představíme rozšíření INETMANET frameworku s názvem ReaSE (Realistic Simulation Environments).

4.2.1 ReaSE

ReaSE je rozšíření pro OMNeT++ v kombinaci s INETMANET frameworkem pro tvorbu realistických síťových prostředí. Rozšíření nabízí hierarchické adresování a směrování spolu s generováním provozu běžícího na pozadí. Dále nabízí simulaci např.: DDOS útoků popřípadě systém detekce těchto útoků. Více informací je k dispozici na oficiálních stránkách projektu: `https://projekte.tm.uka.de/trac/ReaSE`.

5 Praktické příklady

V této kapitole si na konkrétním příkladu předvedeme tvorbu jednoduché sítě. Její funkčnost demonstrujeme vytvořením TCP spojení z klienta na server. Tento text předpokládá, že čtenář provedl instalaci systému podle kapitoly 3.

5.1 Úvod

Pro tvorbu příkladu využijeme grafické uživatelské rozhraní Eclipse, v němž je možno vytvářet síť pomocí psaní zdrojových kódů či v pohodlnějším grafickém režimu.

Pomocí: `File --> New --> OMNeT++ project` si vytvoříme nový projekt v našem případě `ARPTest`. Abychom mohli vytvářet vlastní topologii, tak je potřeba vložit do projektu `NED` soubor. Učiníme tak pomocí `File --> New --> Network Description File(ned)`. Pojmenujeme jej podle názvu projektu (`ARPTest.ned`) a nezapomeneme zatrhnout, aby se jednalo o síťový modul.

Vzhledem k tomu, že budeme chtít využít prvky z INETMANET frameworku, musíme ještě ve vlastnostech projektu zaškrtnout na záložce `Project References` odkaz na INETMANET. Tím se nám v pravé části prostředí přidají prvky z INETMANET v podobě ikon reprezentujících jednoduché a složené moduly.

5.2 Popis topologie

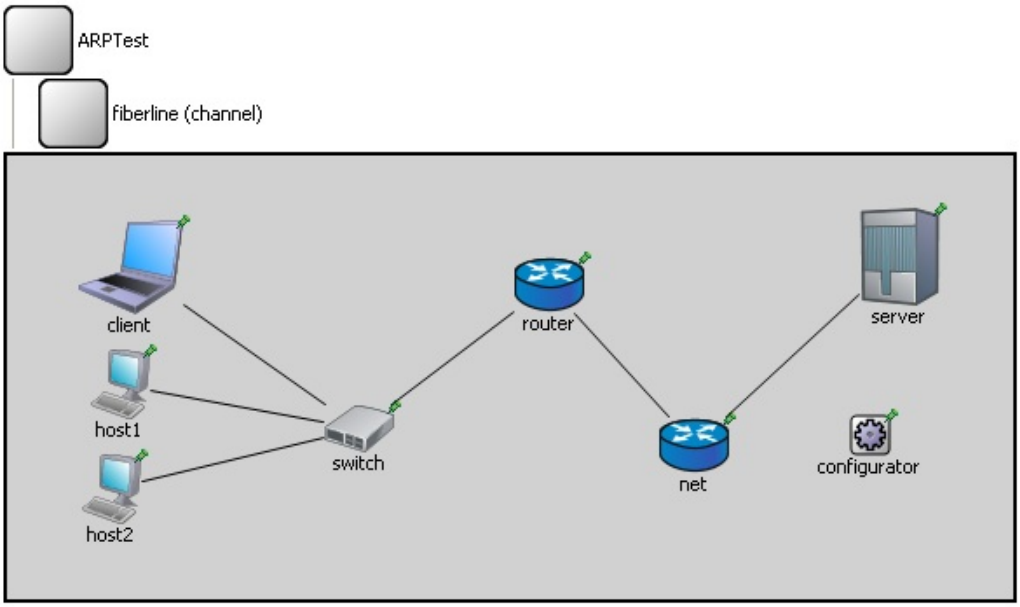
Naše síť je vyobrazena na obrázku 8. Skládá se z 3 počítačů připojených k přepínači pomocí ethernetových 10Mbps linek spolu se směrovačem `router` tvořící jeden segment, dále dvěma směrovači (`router` a `net`) spojených optickou linkou typu point-to-point o rychlosti 512Mbps a stejně tak spojený `server` s `net` směrovačem.

Nastavení parametrů jednotlivých linek (rychlost, zpoždění, chybovost) a dalších vlastností testované topologie se provádí v konfiguračním souboru `omnetpp.ini`. Jeho strukturu řeší další podkapitola.

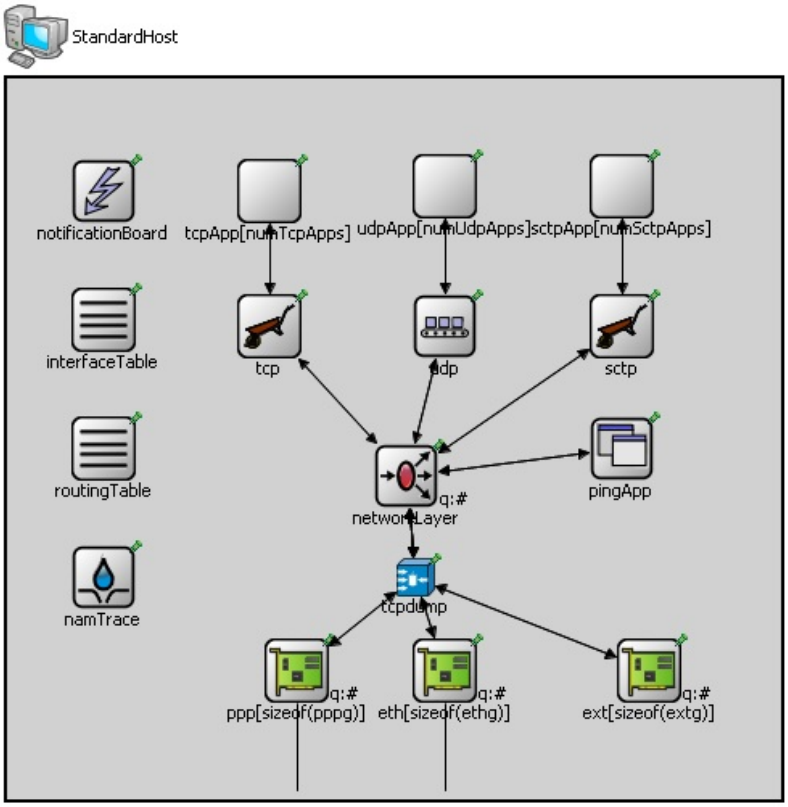
Složený modul směrovače jsme si rozebrali v minulé kapitole. Pojďme se teď krátce podívat na klientský počítač, jehož grafická reprezentace je vidět na obrázku 9. Oproti směrovači, který pracuje na 3. vrstvě, je klient pochopitelně vybaven moduly pro obsluhu vyšších vrstev. Například protokoly 4. vrstvy TCP, UDP, SCTP a také protokolem 3. vrstvy ICMP.

Pro konfiguraci prvků použijeme `FlatNetworkConfigurator`, který nám nakonfiguruje IP adresy všech zařízení (směrovače, klientské stanice) a vyplní také směrovací tabulky.

Při vytváření pomocí grafického rozhraní se nám na pozadí generuje zdrojový kód NED souboru, jehož podoba pro naši topologii je uvedena ve výpise 7. Nakonec ještě dopíšeme parametry pro optiku.



Obrázek 8: Grafické zobrazení testovací topologie



Obrázek 9: Klientský počítač

```
import inet.networklayer.autorouting.FlatNetworkConfigurator;
import inet.nodes.ethernet.EtherSwitch;
import inet.nodes.inet.Router;
import inet.nodes.inet.StandardHost;
import ned.DatarateChannel;

network ARPTest
{
    types:
        channel fiberline extends DatarateChannel
        {
            delay = 1us;
            datarate = 512Mbps;
        }
    submodules:
        client : StandardHost {
            @display("p=71,64;i=device/laptop_l");
        }
        host1: StandardHost {
            @display("p=65,131;i=device/pc");
        }
        host2: StandardHost {
            @display("p=60,191;i=device/pc");
        }
        switch: EtherSwitch {
            @display("p=202,156");
        }
        net: Router {
            @display("p=394,166");
        }
        router: Router {
            @display("p=311,74");
        }
        server: StandardHost {
            @display("p=512,58;i=device/server_l");
        }
        configurator: FlatNetworkConfigurator {
            @display("p=495,160");
        }
    connections:
        client.ethg++ <--> switch.ethg++;
        switch.ethg++ <--> host1.ethg++;
        switch.ethg++ <--> host2.ethg++;
        router.ethg++ <--> switch.ethg++;
        router.pppg++ <--> fiberline <--> net.pppg++;
        server.pppg++ <--> fiberline <--> net.pppg++;
}
```

Výpis 7: Vygenerovaný popis topologie

5.3 Inicializace modulů

Jak jsme se již dříve zmínili, nastavení parametrů simulace se provádí v konfiguračním souboru `omnetpp.ini`. Pojďme si jej tedy vytvořit. `File --> New --> Initialization File(INI).` `INI` soubor, stejně jako `NED`, se dá modifikovat v grafickém či v textovém režimu.

Nastavíme parametry TCP spojení jak je vidět na výpise 8. Jádru simulačního systému jsme tedy přikázali simulovat síť ARPTest. U klienta jsme nastavili jednu TCP aplikaci, která se bude snažit navázat spojení se serverem, který bude poslouchat na portu 1000. Dále pak některé další parametry.

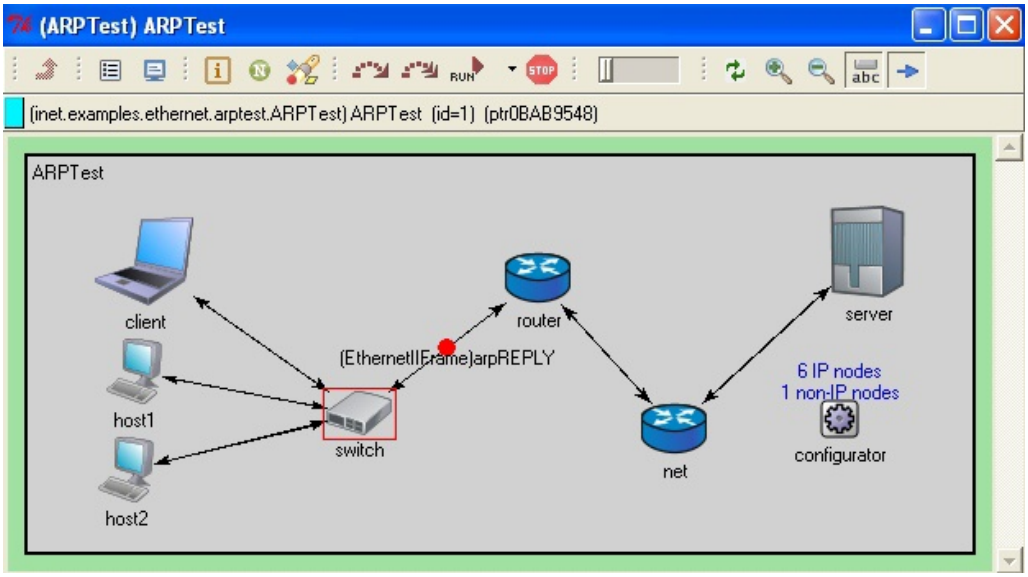
```
network = ARPTest

# TCP apps
**.client.numTcpApps = 1
**.client.tcpAppType = "TCPSessionApp"
**.client.tcpApp[*].active = true
**.client.tcpApp[*].address = ""
**.client.tcpApp[*].port = -1
**.client.tcpApp[*].connectAddress = "server"
**.client.tcpApp[*].connectPort = 1000
**.client.tcpApp[*].tOpen = 1.0s
**.client.tcpApp[*].tSend = 1.1s
**.client.tcpApp[*].sendBytes = 1MB
**.client.tcpApp[*].sendScript = ""
**.client.tcpApp[*].tClose = 0

**.server.numTcpApps = 1
**.server.tcpAppType = "TCPEchoApp"
**.server.tcpApp[0].address = ""
**.server.tcpApp[0].port = 1000
**.server.tcpApp[0].echoFactor = 2.0
**.server.tcpApp[0].echoDelay = 0

# TCP settings
**.tcp.sendQueueClass = "TCPVirtualDataSendQueue"
**.tcp.receiveQueueClass = "TCPVirtualDataRcvQueue"
```

Výpis 8: INI soubor – TCP parametry

Z těch zajímavějších řádků tu máme ještě nastavení síťových karet na stanicích na 10Mbit, jak je vidět na výpise 9.

```
# Ethernet NIC configuration
**.eth[*].mac.txrate = 10Mbps
**.eth[*].mac.duplexEnabled = true
```

Výpis 9: INI soubor – nastavení NIC



Obrázek 10: Cesta ARP datagramu

5.4 Ověření funkčnosti

Nyní je tedy vše potřebné nastaveno a můžeme přejít ke spuštění simulace. Jak si lze povšimnout z výpisu 10, `FlatNetworkConfigurator` nastavil na začátku všem aktivním prvkům IP adresy a směrovací tabulky.

```
cTopology found 7 nodes
  client=192.168.0.1 has only one (non−loopback) interface, adding default route
  host1=192.168.0.2 has only one (non−loopback) interface, adding default route
  host2=192.168.0.3 has only one (non−loopback) interface, adding default route
  server=192.168.0.6 has only one (non−loopback) interface, adding default route
  from net=192.168.0.4 towards client=192.168.0.1 interface ppp0
  from router=192.168.0.5 towards client=192.168.0.1 interface eth0
  from net=192.168.0.4 towards host1=192.168.0.2 interface ppp0
  from router=192.168.0.5 towards host1=192.168.0.2 interface eth0
```

Výpis 10: Nastavení IP adres a směrovacích informací (zkrácená verze)

Dále pak klient vytváří TCP spojení na server ze zdrojového portu 1025 na IP adresu 192.168.0.6 na port 1000, jak je vidět na následujícím výpise:

```
TCP connection created for (cMessage)ActiveOPEN
Connection <unspec>:−1 to <unspec>:−1 on app[0],connId=0 in INIT (ptr=0x0xcbe2308)
App command: OPEN_ACTIVE
Assigned ephemeral port 1025
OPEN: <unspec>:1025 −−> 192.168.0.6:1000
Sending: .1025 > .1000: SYN 250000:250000(0) win 14336
Transition : INIT −−> SYN_SENT (event was: OPEN_ACTIVE)
```

Výpis 11: Vytvoření TCP spojení

Známe tedy cílovou IP adresu, ne však fyzickou adresu dalšího zařízení v cestě (v našem případě router). Dojde tedy k použití ARP protokolu, kdy je broadcastově poslán dotaz: „Kdo má IP adresu 192.168.0.6?“. V tomto případě odpoví router (viz. obrázek 10) a na zařízení odešle datagram ARPReply viz. následující výpis:

```
ARP packet (ARPPacket)arpREPLY arrived:
ARP_REPLY src=192.168.0.6 / 0A-AA-00-00-00-08 dest=192.168.0.1 / 0A-AA
    -00-00-00-01
Updating ARP cache entry: 192.168.0.6 <--> 0A-AA-00-00-00-08
Sending out queued packet (IPDatagram)SYN
Discarding packet
```

Výpis 12: Zpracování ARP reply

Ten klient zpracuje a uloží si záznam do ARP cache. A nyní je schopen zahájit TCP spojení vysláním datagramu SYN.

6 Závěr

Tato práce byla zadána v květnu roku 2008, kdy byla na světě verze 3.3 z roku 2006. V té době byl ještě aktuální INET framework a dnes již ukončený projekt Mobility framework. Práce na bakalářské práci tedy započaly v září loňského roku. Zpočátku byly celkem velké problémy s instalací, která vyžadovala dalších cca 10 úkonů pro správnou funkčnost.

Vše tedy vyřešilo nástup nové verze ke konci února 2009. Ta přinesla nové vývojové prostředí založené na Eclipse a podstatně zjednodušila instalaci. Předělán byl také NED jazyk, u kterého byla upravena syntaxe, vylepšeno spojování modulů a zavedení nových komunikačních kanálů. Stejně tak byl inovován INET framework a došlo k jeho sloučení s Mobility frameworkem. Když se k tomu ještě přidá výborná dokumentace dostupná na hlavních stránkách projektu, tak tu máme celkem mocný nástroj pro simulaci počítačových sítí.

Právě dokumentace byla výrazně inovována. Ať už se jedná o 2 publikace Andrease Vargy **OMNeT++: Discrete Event Simulation System – User manual** [1] a **OMNeT++ User Guide** [2], nebo velmi dobře zpracované hlavní stránky [3], kde se dá najít spousta rad a informací. Nakonec bych chtěl ještě vyzvednout stránky INEMANET frameworku [4], kde se nachází podrobný popis všech modulů spolu s jejich parametry.

Už na počátku jsem také narazil na velmi zajímavé diplomové práce, bohužel napsané pro verzi 3.3. Jednalo se o práci pana Ing. Jiřího Šádka, který se zabýval rozšířením INET frameworku o protokol SNMP v diplomové práci **SNMP správa pro modely OMNeT** [11]. Na jeho práci potom v roce 2009 navázal Ing. Jan Mach, který systém rozšířil tak, aby jej bylo možno použít pro výuku. Tyto informace publikuje ve své diplomové práci s názvem **Emulátor počítačové sítě** [12].

Další práce od Ing. Františeka Bernharda **Přístupové metody WIFI** [9] zkoumala analýzu chování a komunikace mezi jednotlivými uzly v současnosti velmi rozšířených bezdrátových sítí dle standardu 802.11 a to ve variantách a, b, g. Na základě této analýzy byly sestaveny a implementovány modely umožnující simulaci chování jednotlivých uzlů i větších celků a jejich interakci mezi sebou.

Nakonec bych ještě zmínil pana Ing. Jana Michka a jeho diplomovou práci z roku 2008 s názvem **Emulátor počítačové sítě** [10]. V této práci se úspěšně snažil navrhnout rozhraní, které umožní konfiguraci simulovaných síťových prvků z modelové sady INET framework. Rozhraní by mělo nabízet uživateli stejné možnosti, jako kdyby měl k dispozici reálné terminály systémů Linux a IOS od firmy Cisco.

Během tvorby mé práce jsem si také zjišťoval mezi studenty vzdělanými v oblasti počítačových sítí, jaké mají zkušenosti se simulátory a jestli znají simulační prosředí OMNeT++. Většinou jsem se setkal s lidmi, kteří už někdy v minulosti pracovali s Packet Tracerem od firmy Cisco, ale zrovna nadšeni nebyli. Když jsem se jich zeptal na OMNeT++, tak ani v jednom případě o tom nikdo nic nevěděl. Je pravdou, že ti „šťastnější“ mají možnost pracovat se skutečnými zařízeními (třeba v rámci Cisco akademie), což je samozřejmě ta nejlepší varianta, nicméně ti ostatní „méně šťastní“ mohou spoustu věcí poznat a vyzkoušet v simulačním prostředí.

Proto tedy vznikla moje práce, aby dala možnost všem těmto zájemcům proniknout do nástrah tohoto simulačního systému bez zbytečného tápání. Běžný uživatel simulátoru tedy potřebuje alespoň částečně pochopit základní pojmy systému, nainstalovat jej, zjistit, jak se tvoří topologie a generují zprávy. Dále už je pak na něm, jak moc se bude chtít zanořit do vod OMNeT++ a INETMANET frameworku.

7 Reference

[1] Varga, Andreas. *OMNeT++: Discrete Event Simulation System – User manual*. Version 4.0, 2008. 361 s.

[2] Varga, Andreas. *OMNeT++ User Guide*. 2008. 100 s.

[3] Contributing authors. *OMNeT++ Discrete Event Simulation System – community site* [online]. c2003, last revision 28th of April 2009 [cit. 2009-05-02]. http://www.omnetpp.org/.

[4] Contributing authors. *INET framework for OMNeT++ 4.0* [online]. c2008, last revision 16th of March 2009 [cit. 2009-05-02]. http://inet.omnetpp.org/.

[5] Contributing authors. *Model documentation for INET framework* [online]. c2008, last revision 12th of March 2009 [cit. 2009-05-02]. http://inet.omnetpp.org/doc/INET/neddoc/index.html.

[6] Contributing authors. *OMNeT++ User Community Wiki* [online]. c2005, last revision 23th of April 2009 [cit. 2009-05-02]. http://inet.omnetpp.org/doc/INET/neddoc/index.html.

[7] Contributing authors. *Realistic Simulation Environments for OMNeT++* [online]. c2008, last revision 7th of April 2009 [cit. 2009-05-02]. https://projekte.tm.uka.de/trac/ReaSE.

[8] Contributing authors. *Wikipedia – The free encyclopedia* [online]. c2001, last revision 30th of April 2009 [cit. 2009-05-02]. http://en.wikipedia.org/.

[9] Bernhard, František. *Přístupové metody WIFI*. Praha, 2007. 84 s. Diplomová práce na Fakultě elektrotechniky ČVUT na katedře počítačů. Vedoucí diplomové práce Doc. Ing. Jan Janeček, CSc.

[10] Michek, Jan. *Emulátor počítačové sítě*. Praha, 2008. 116 s. Diplomová práce na Fakultě elektrotechniky ČVUT na katedře počítačů. Vedoucí diplomové práce Ing. Jan Kubr.

[11] Šádek, Jiří. *SNMP správa pro modely OMNet*. Praha, 2007. 76 s. Diplomová práce na Fakultě elektrotechniky ČVUT na katedře počítačů. Vedoucí diplomové práce Doc. Ing. Jan Janeček, CSc.

[12] Mach, Jan. *Emululátor počítačové sítě*. Praha, 2009. 83 s. Diplomová práce na Fakultě elektrotechniky ČVUT na katedře počítačů. Vedoucí diplomové práce Doc. Ing. Jan Janeček, CSc.

A Obsah CD

Bakalářská práce obsahuje vložené CD, na němž je možno najít část použité literatury a tento text v elektronické podobě ve formátech PDF a LaTeX. Dále se zde nachází instalační soubory pro Linux, Windows a archiv obsahující INETMANET framework.

Adresář	**Podadresář**	**Obsah**
doc	bak	Text práce, formát PDF a LaTeX
doc	lit	Část použité literatury
src	omnet	Instalační balík pro Linux a Windows
src	inet	Archiv s INETMANET frameworkem

Tabulka 1: Obsah CD